



TITLE:

# <技術報告>マイコンを使った風速計測

AUTHOR(S):

米田, 格

---

CITATION:

米田, 格. <技術報告>マイコンを使った風速計測. 技術室報告 2018, 19

ISSUE DATE:

2018-06

URL:

<http://hdl.handle.net/2433/233573>

RIGHT:

# マイコンを使った風速計測

米田 格

# 計測のための最低条件

- 風速データが取れること
- バッテリーで動くこと
- 安価であること
- (無線ネットワークが使用できるのであれば)  
データサーバにデータを送ることができること

# マイコンのメリット

- 小型

- 省電力

デスクトップパソコン	50～150W
ノートパソコン	20～30W
raspberry pi (マイコン)	7W

- 安価

パソコン	数万円
マイコン	数千円

# マイコンボード

- マイコンと信号の入出力に必要な部品を合わせた、マイコンボードを利用することで手軽に使える
- 様々なセンサーや機器を気軽に追加できる



# マイコンボードの種類



Arduino

AD変換も内蔵されており、手軽に計測が出来る



Raspberry Pi

Linuxベースのためサーバとしても使える

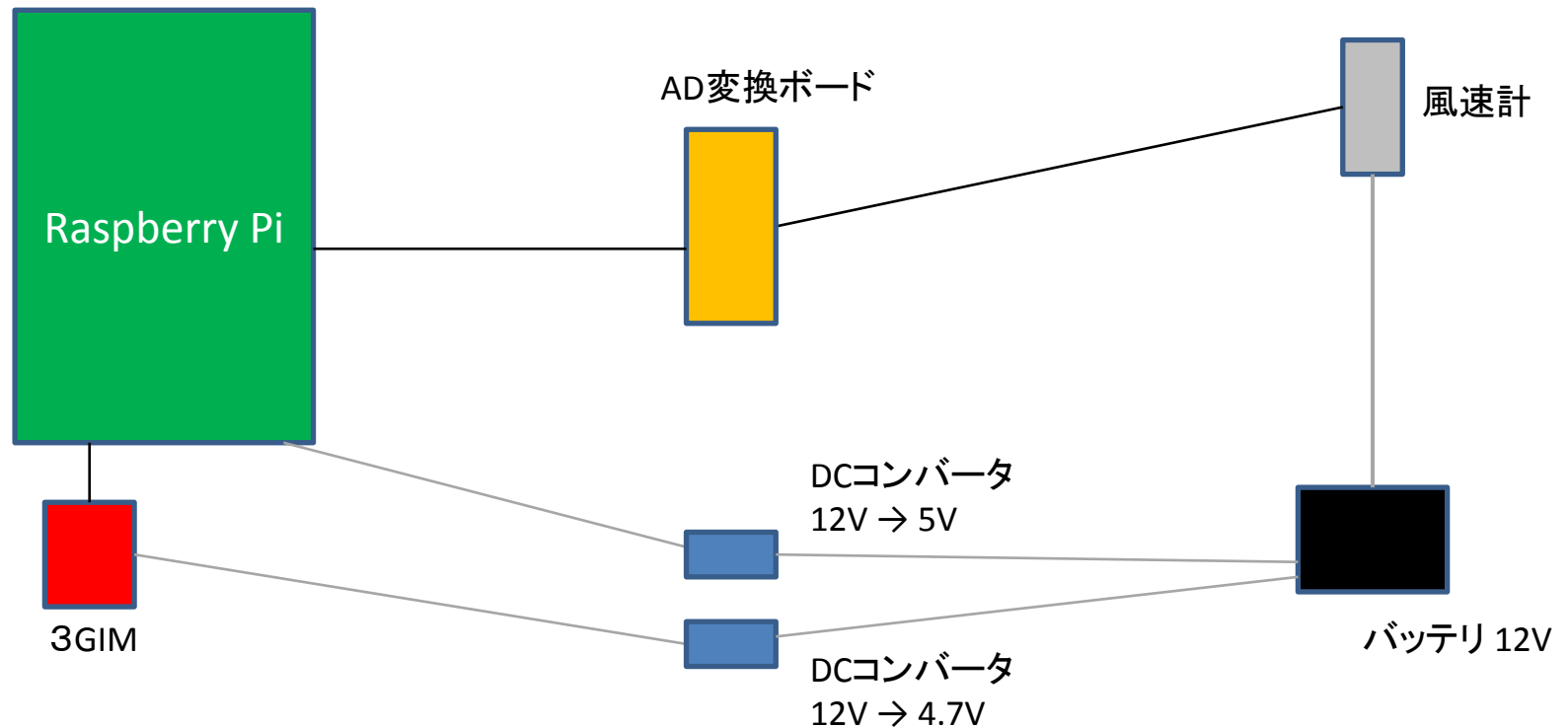


GR-PEACH

内蔵メモリが大きいいため大きなデータを扱える

# 構成図

今回はRaspberry Piを使用

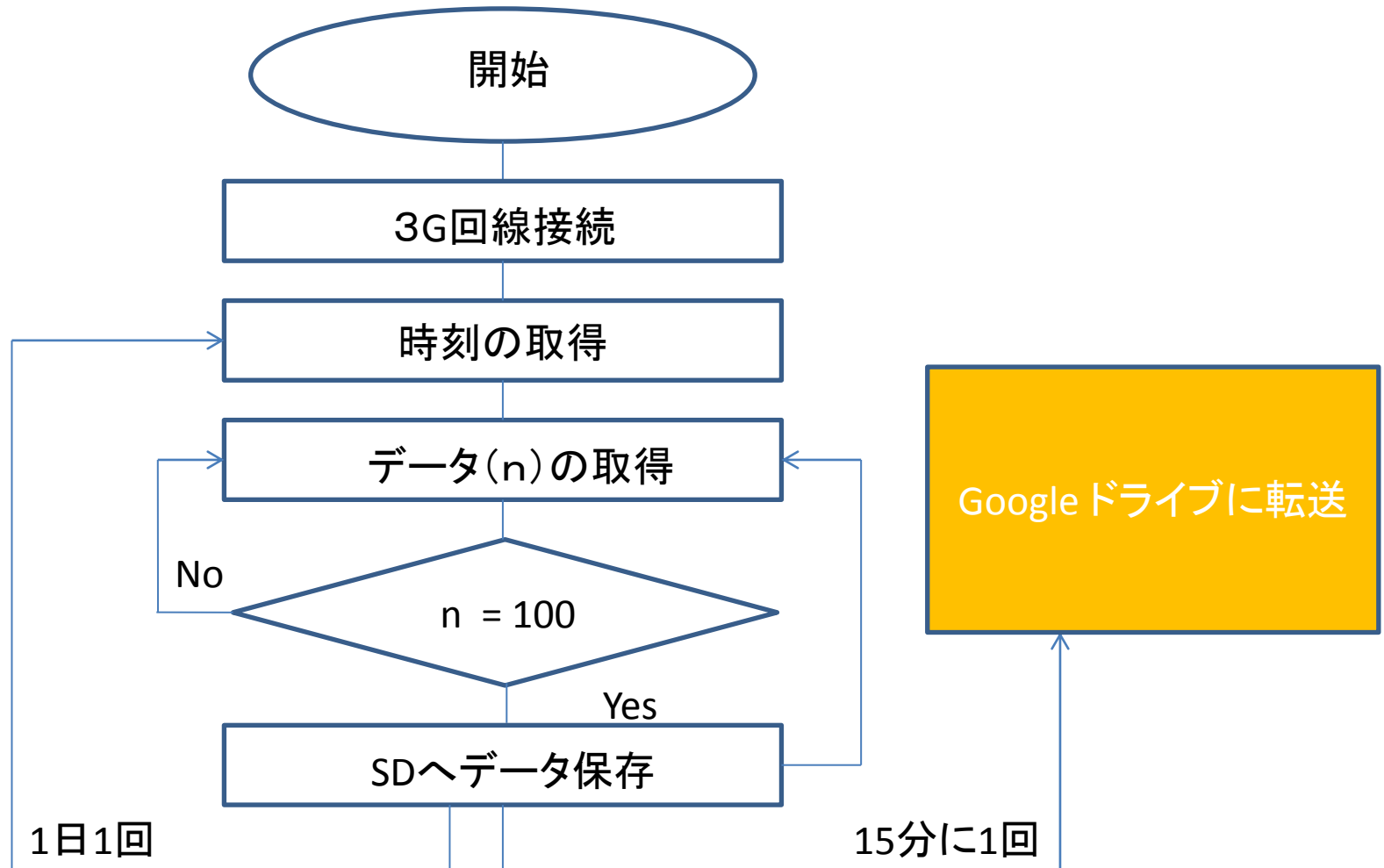


# スペック

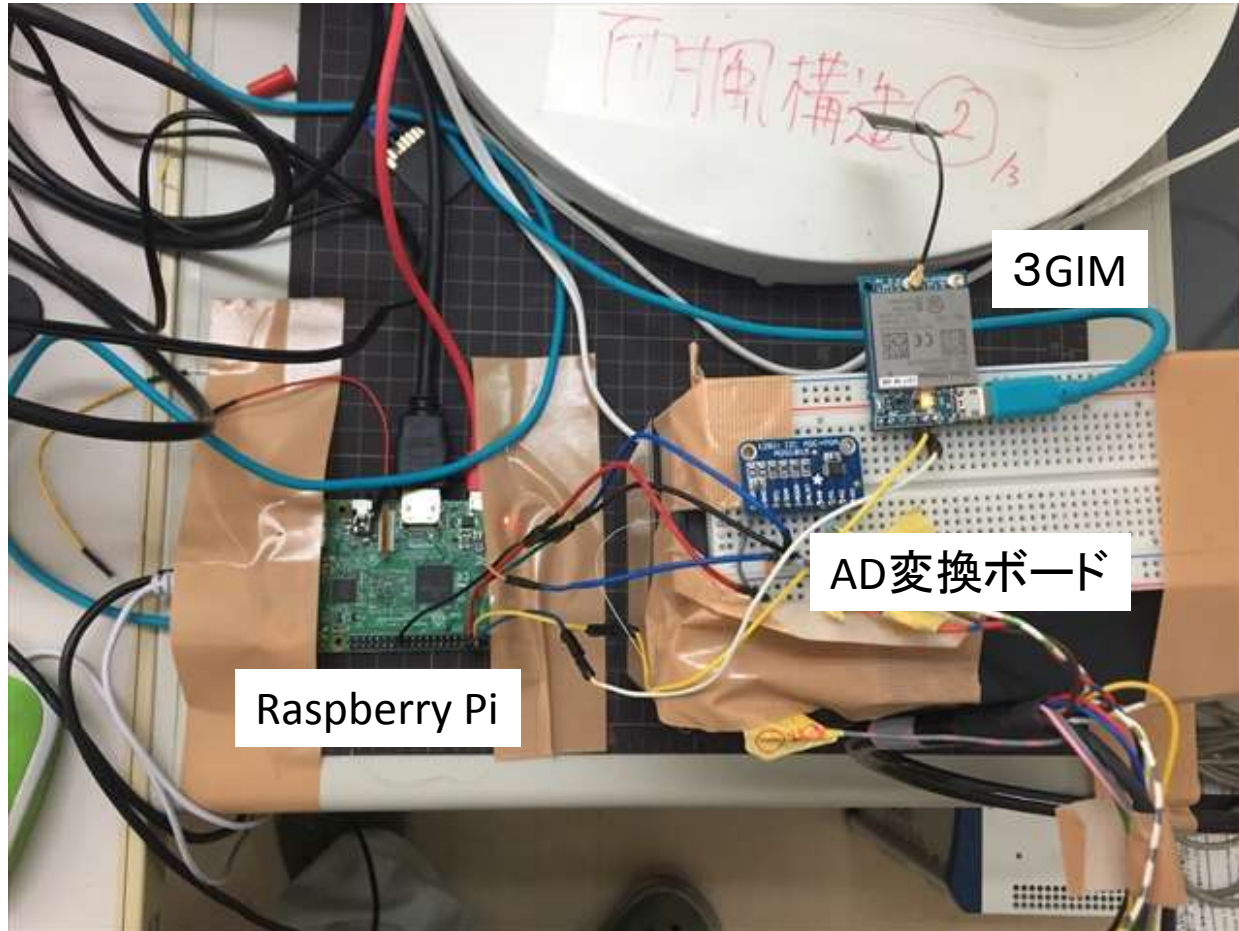
- サンプルング周波数 100Hz(出力は1秒平均)
- 量子化ビット数 16bit
- 風速90m/sまで計測可能



# データ測定のプロフローチャート(予定)



# 室内での計測テスト



# 問題点1

書き込み時間が安定しない



SD書き込み時に時間がかかってしまい、  
計測が予定の時間より遅れてしまっている

MS Pゴシック			
B I U			
クリップボード			
フォント			
時刻	風速	風向	
1	0.00	3.416	0.016
2	0.01	3.434	0.013
3	0.03	2.83	0.016
4	0.04	2.094	0.153
5	0.06	2.309	0.061
6	0.07	2.283	0.032
7	0.08	2.055	0.051
8	0.10	1.82	0.155
9	0.11	2.238	0.074
10	0.13	2.459	0.024
11	0.14	2.962	0.021
12	0.16	3.082	0.021
13	0.17	3.514	0.014
14	0.19	3.35	2.179
15	0.20	1.957	0.153
16	0.22	1.742	0.036
17	0.23	1.984	0.018
18	0.25	2.04	0.112
19	0.26	2.029	0.169
20	0.27	2.426	0.046
21	0.29	2.662	0.026
22	0.30	2.599	0.024
23	0.32	3.079	0.021
24	0.33	3.605	0.761
25	0.35	2.282	0.127
26	0.36	1.916	0.119
27	0.38	1.956	0.049
28	0.39	1.317	0.015
29	0.41	1.242	0.089
30	0.42	2.089	0.146
31	0.43	2.747	0.039
32	0.45	3.056	0.026
33	0.46	2.979	0.018
34	0.48	3.173	0.014
35	0.49	3.232	0.04
36	0.51	2.077	0.149
37	0.52	1.571	0.074
38	0.54	1.624	0.02
39	0.55	2.003	0.019
40	0.57	1.885	0.155
41	0.58	2.24	0.109

# 解決策

プログラムをマルチスレッド化し、SDが書き込まれていても時間になったら計測が始まるようにした

```
#!/path/to/python3
# -*- coding: utf-8 -*-
import Adafruit_ADS1x15
import time
from datetime import datetime
from concurrent.futures import ThreadPoolExecutor
```

```
GAIN = 1
DR = 250
sampling = 100
```

```
adc = Adafruit_ADS1x15.ADS1015()
```

```
def fileWrite(time,value1,value2):
    value1 = ((75+47)/47)*4.096*(value1/sampling)/2048
    value2 = ((75+47)/47)*4.096*(value2/sampling)/2048
    now = datetime.now()
    datafile = now.strftime("%y%m%d%H") + ".csv"
    f = open("/home/pi/vanuatou_data/" + datafile,'a')
    f.writelines(str(time) + "," + str(value1) + "," + str(value2) + "\n")
```

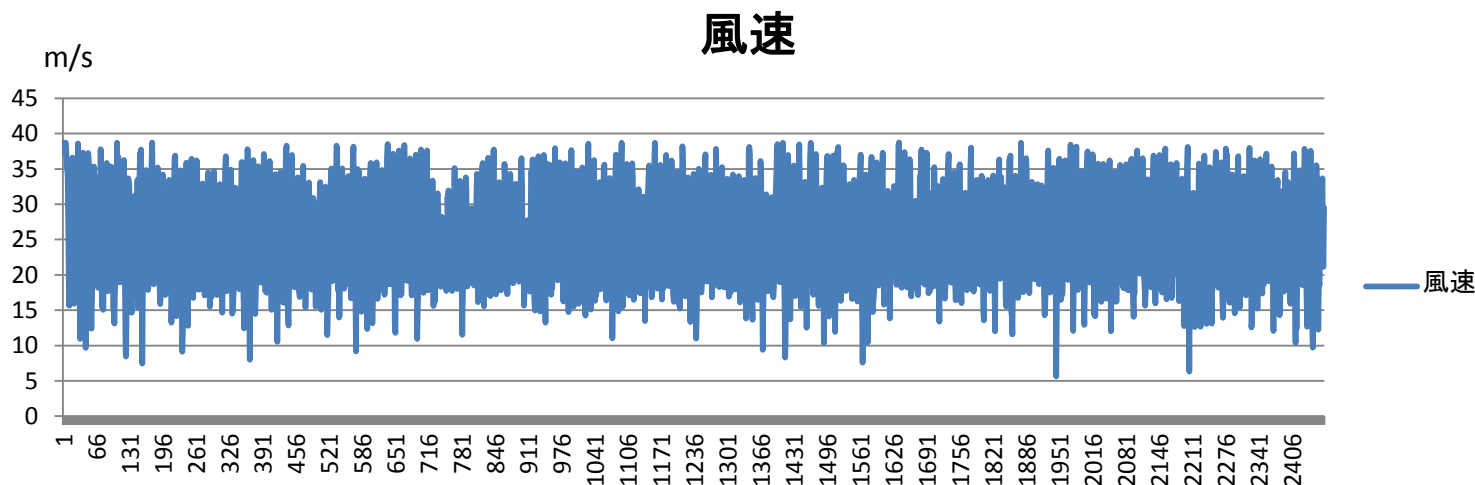
```
pool = ThreadPoolExecutor(4)
while True:
    vspeed = 0
    vdeg = 0
    mmss = time.strftime("%M:%S,")
    for i in range(sampling):
        vspeed = vspeed + adc.read_adc(0, gain=GAIN)
        vdeg = vdeg + adc.read_adc(1, gain=GAIN)
        time.sleep(0.3/sampling)
```

```
pool.submit(fileWrite,mmss,vspeed,vdeg)
```

SDへの書き込みはスレッドを分けて処理

# 問題点2

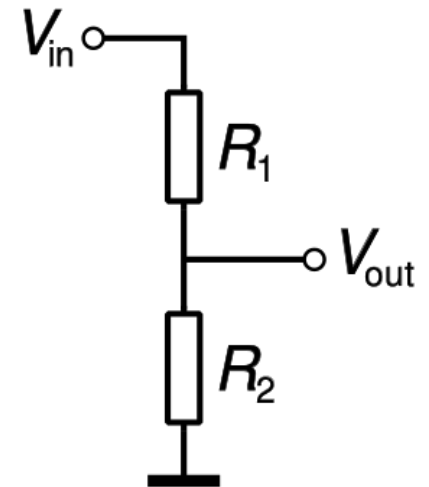
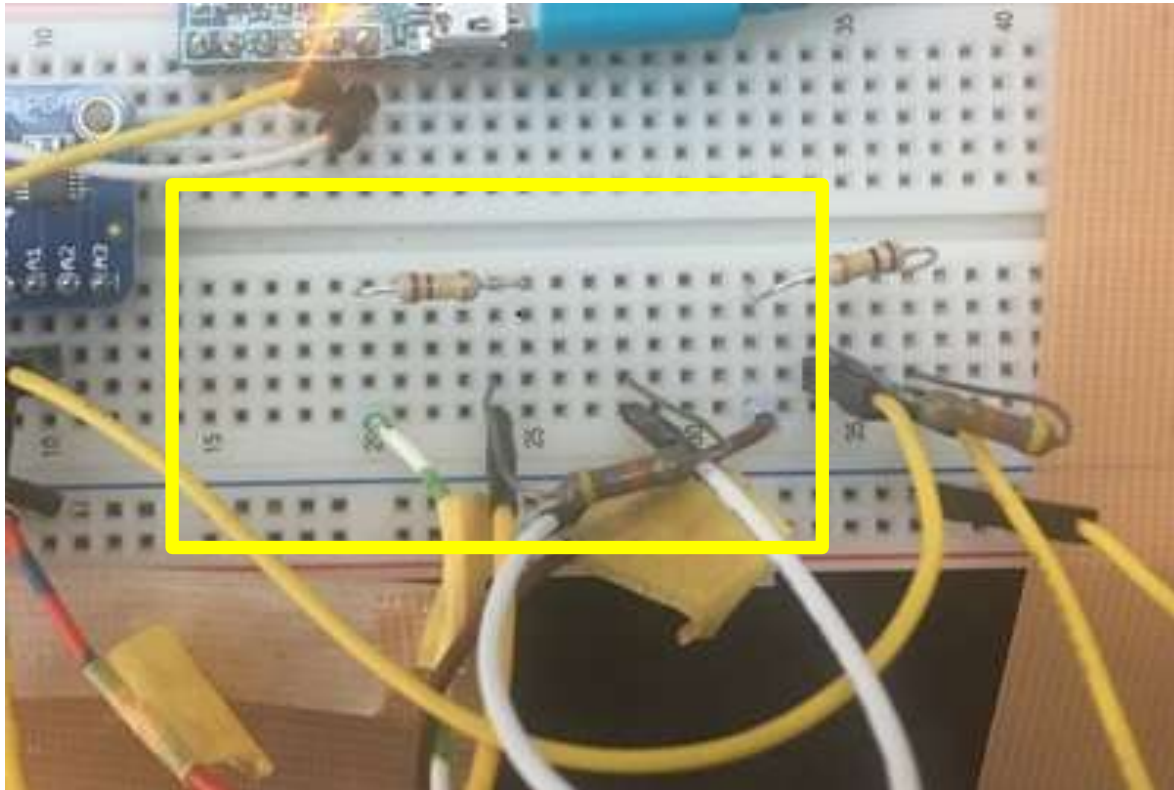
- ノイズが大きい



5V以上の電圧が入っている  
(raspberry Piが計測できるのは5Vまで)

# 解決策

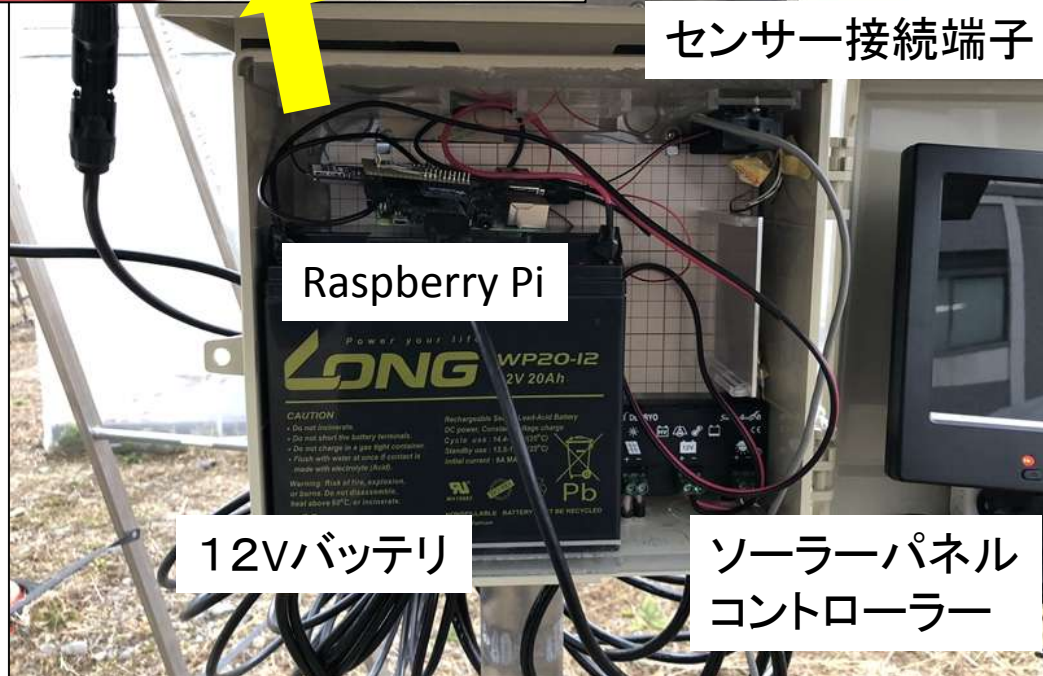
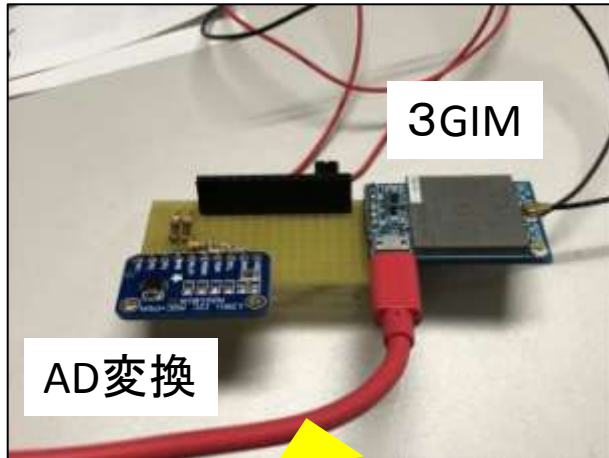
分圧回路を利用し、風速計からの電圧を5V以下になるよう調整した



$$V_{\text{out}} = \frac{R_2}{R_1 + R_2} \cdot V_{\text{in}}$$



# 室外での計測テスト



# 室外テストでの問題点1

- リモートアクセスが出来ない



グローバルIPが振られていない(SIM会社に依存)

```
RA packets 80 bytes 37050 (36.1 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 80 bytes 37050 (36.1 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

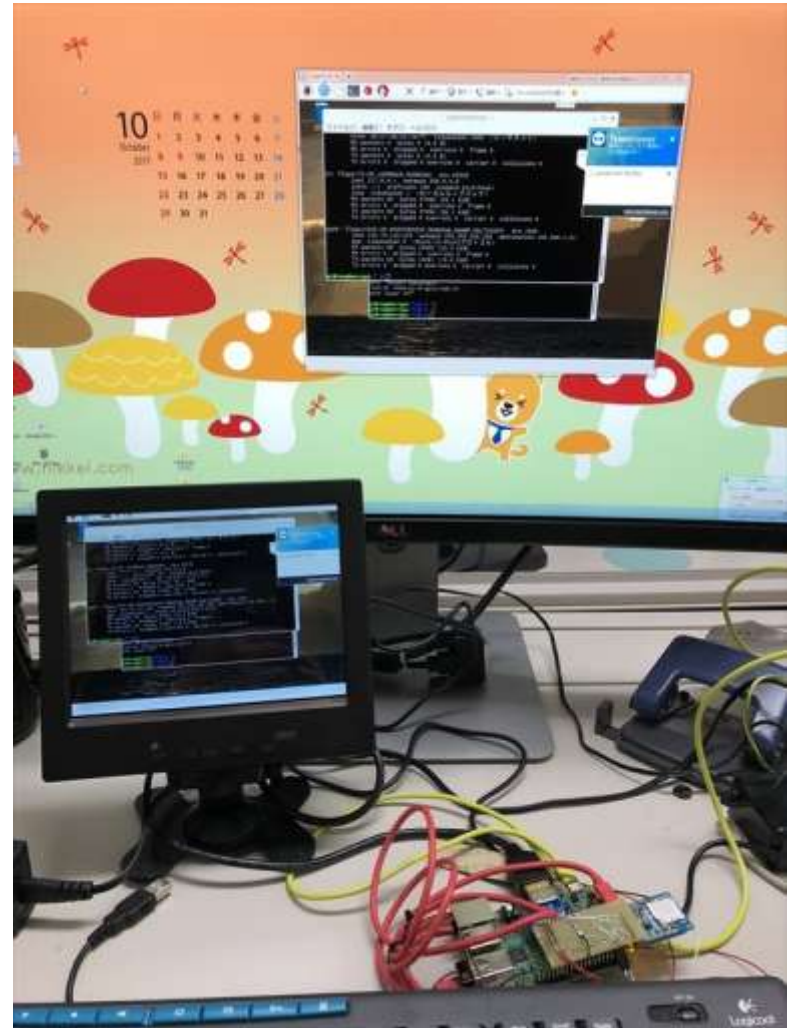
ppp0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
inet 100.79.143. netmask 255.255.255.255 destination 192.200.1.21
ppp txqueuelen 3 (Point-to-Pointプロトコル)
RX packets 58 bytes 24033 (23.4 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 68 bytes 13381 (13.0 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pi@raspberrypi:~$
echo network disconnect
/bin/sh /home/pi/sh/gpio_high.sh
echo "power off"
```



# 解決策

- IPではなくアカウントでリモート接続できる  
TeamViewerを利用



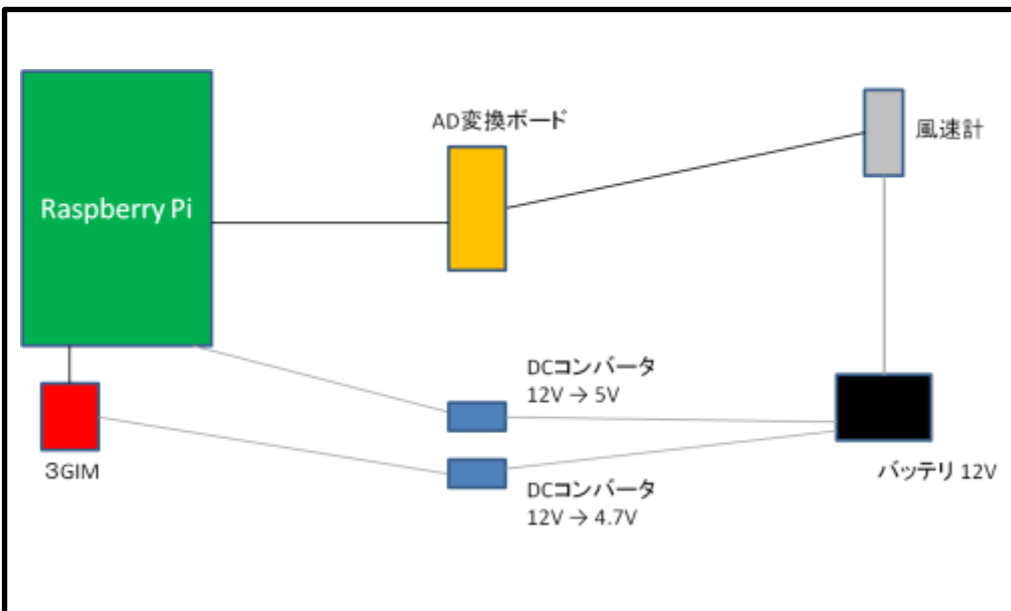
# 室外テストでの問題点2

Raspberry Piの動作が安定しない

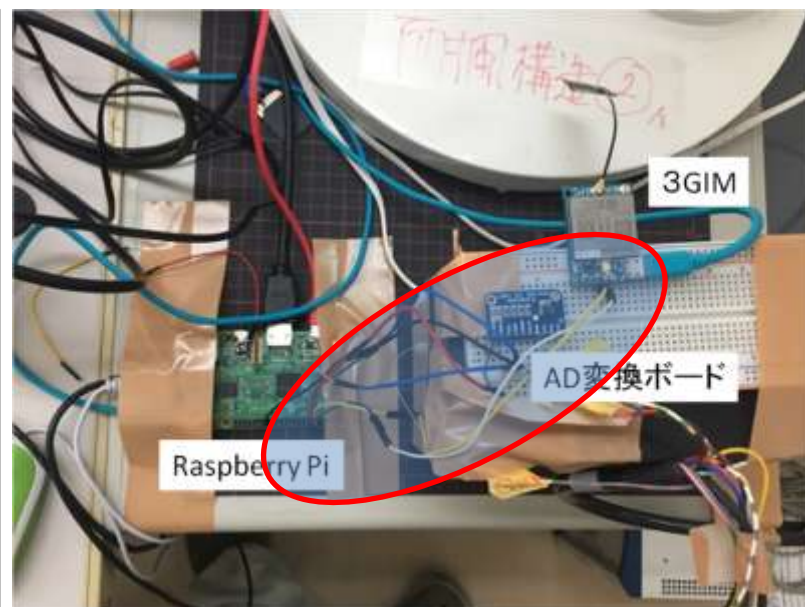


Raspberry Piから3G用の基盤に電源を取っていたため

従来の配線



テスト時の配線



# 解決策

## 3G用の基盤の電源を別に用意した

TA Brain 3GIM 2.1

### 5. 3GIM V2.1 のピンコネクタ配置

ピン番号	名称	機能など
#1	PWR_ON	電源のON/OFF制御(開放または0:LOWでON、1:HIGHでOFF)
#2	RX	UARTインタフェース(RX) : 相手方のTxに接続
#3	TX	UARTインタフェース(TX) : 相手方のRxに接続
#4	IOREF	ロジック電圧(任意、通常は1.8V~5V間で利用)
#5	VCC (+)	電源電圧(3.3~4.2V) ※接続時に3.2V以下にならないこと
#6	GND (-)	グラウンド

【補足説明】

- #1ピンは「1」とシルク印刷されている側のピンです。
- #1ピンの「HIGH」は、ロジック電圧(IOREF)とします。
- 「VCC(+)」ピンから電源を供給する場合は、PWR\_ONの状態によらず、常にONとなります

※ #5 (VCC) から電源供給必要  
電源電圧 (3.3~4.2V : 200mA以上)  
<#1で電源ON/OFF制御>

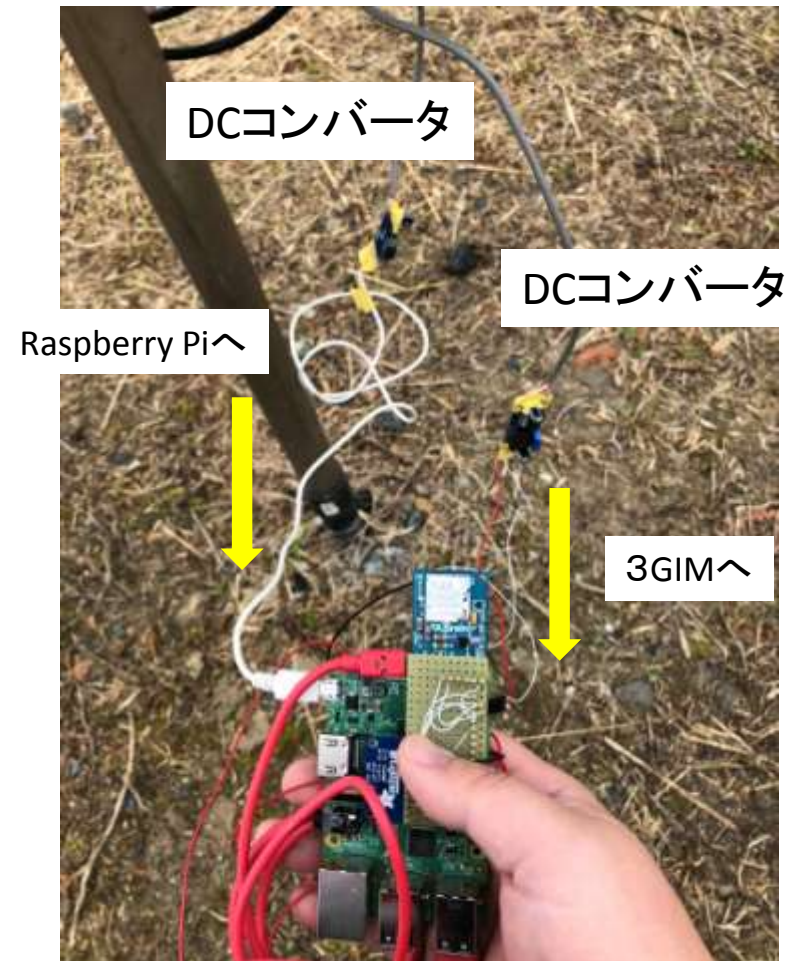
【注意】 Arduinoの外部出力電源の3.3Vでテストしましたが、一部のArduinoでは稼働しないことが分りました。供給電流が低いために通信できないものと思われます。(補足：場合によってはPC側の電源供給不足問題もあります)



9

Open Wireless Alliance

### 3GIM説明書



# 室外テストでの問題点3

5日でバッテリーが落ちてしまった



3G回線に常時接続されているため、必要以上に電力を消耗？



# 解決策

データをgoogleドライブに転送するときだけ、  
3G回線に接続(転送後3G用基盤の電源を切断するようにした)

ABrain

## 5. 3GIM V2.1 のピンコネクタ配置

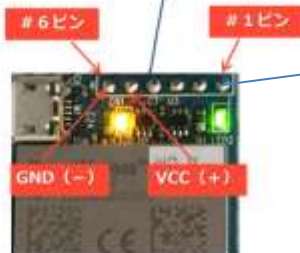
ピン番号	名称	説明
#1	PWR_ON	電源のON/OFF制御(開放または0:LOWでON、1:HIGHでOFF)
#2	KX	UARTインタフェース(KX): 相手のTxに接続
#3	TX	UARTインタフェース(TX): 相手のRxに接続
#4	IOREF	ロジック電圧(任意、通常は1.8V~5V間で利用)
#5	VCC(+)	電源電圧(3.3~4.2V) ※接続時に3.2V以下にならないこと
#6	GND(-)	グラウンド

### 【補足説明】

- #1ピンは「1」とシルク印刷されている側のピンです。
- #1ピンの「HIGH」は、ロジック電圧(IOREF)とします。
- 「VCC(+)」ピンから電源を供給する場合は、PWR\_ONの状態によらず、常にONとなります

※ 5 (VCC) から電源供給必要  
電源電圧 (3.3~4.2V: 200mA以上)  
< #1で電源ON/OFF制御 >

【注意】 Arduinoの外部出力電源の3.3Vでテストしましたが、一部のArduinoでは稼働しないことが分かりました。  
供給電圧が低いために通信できないものと思われます。  
(補足: 場合によってはPC側の電源供給不足問題もあります)

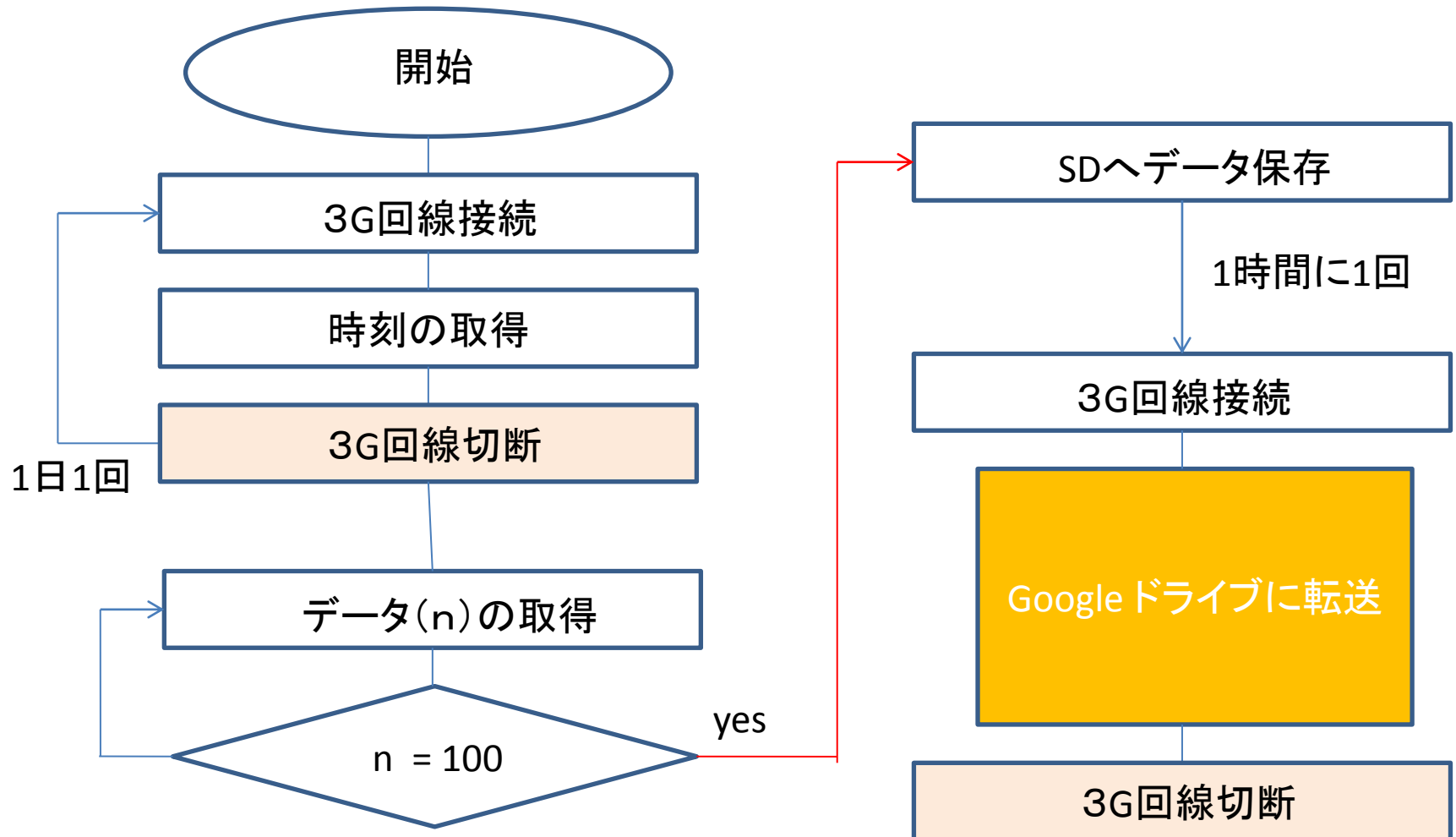


```
pi@raspberrypi:~/sh $ more transport.sh
#!/bin/sh

/bin/sh /home/pi/sh/gpio_low.sh
#echo "power on"
sleep 1m
sudo systemctl start wvdial.service
#echo "network connect"
sleep 20s
#echo "transport now"
/home/pi/bin/gdrive sync upload /home/pi/vanuatu_data/ 1Ce8p77Wf5Jn
dc >> /home/pi/ana.log 2>> /home/pi/ana_err.log
#echo "finish"
sudo systemctl stop wvdial.service
sleep 10s
#echo "network disconnect"
/bin/sh /home/pi/sh/gpio_high.sh
#echo "power off"
pi@raspberrypi:~/sh $
```



# 最終的なフローチャート



# 今後の課題

- 2G(GIM)回線でも使用できるか確認
- サンプリングを安定させる
- 別のマイコンボードを使った計測システムの作成